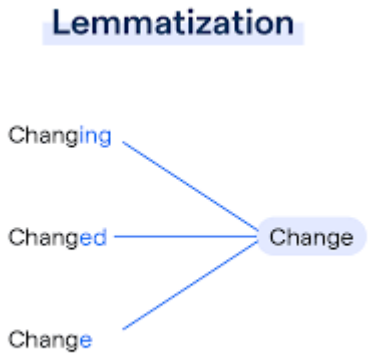
6. Lemmatization

Another preprocessing steps in Natural Language Processing(NLP) used to reduce words to their base/root form.



Overview

Process of reducing a word to its base or dictionary form(lemma).
This process consider the context and morphological analysis of words.

Unlike stemming, **lemmatization aim to remove inflectional endings only and return the base or dictionary form of a word, which is a valid word.**

> **Inflectional Endings**
>
> Suffixes added to the end of words to change or clarify their meaning.
> They can indicate tense, number, possession or comparison.
>
> Ex:
>
> 1. -s / es : Indicate singularity or plurality of a noun
> 2. -ing / -ed : Indicate the tense of a verb
> 3. -'s : a possessive ending (Kalani -> Kalani's)
> 4. -er : comparative ending(Big -> Bigger)
> 5. . -est : Superlative ending(Big -> Bigger -> Biggest)

Ex:

1. Running -> Run
2. Cats -> Cat
3. Better -> Good

Popular tools for Lemmatization,

1. WordNet Lemmatizer
2. spaCy

1. WordNet Lemmatizer

Part of the NLTK(Natural Language Toolkit) library.
Uses the WordNet lexical database to find the base form(lemma) of words.
Considers the context provided by **part-of-speech**(POS) tags to perform accurate lemmatization.

Benefits,

- Context-Awareness
- Rich lexical database
  A database consist of a wide range of words and their lemmas.
- Easy to use within the NLTK framework, which is widely adopted in the NLP community.

Drawbacks,

- Performance Issues: (not much of a consideration)
  Can be slower compared to other lemmatizers due to its reliance on the extensive WordNet database.
- Efficiency: (not much of a consideration)
  Requires POS tags for better accuracy, which adds an extra preprocessing step.

Use Cases,
Suitable for

1. Research purposes where accuracy is crucial and processing time is less of a concern.
2. projects that benefit from the extensive lexical information provided by WordNet.

2. spaCy Lemmatizer

spaCy is a modern NLP library provides an efficient and high-performance lemmatizer.
This lemmatizer is integrated into its tokenization, POS tagging, and dependency parsing modules.

Benefits,

- Optimized for speed and efficiency.
- Integrated Pipelines
  Provides seamless integration with other spaCy components such as tokenization and POS tagging.

- Simplifies the process with minimal setup and high-level API.
  Drawbacks,
- Limited lexical Database
  Might not be as extensive as WordNet for some niche or less common words.
- Requires downloading and using spaCy models, which can be large.

Use Cases,
Ideal for,

1. Real-time applications and production systems where performance is critical.
2. Large-Scale Text Processing

Summary

Feature	WordNet	spaCy
Accuracy	high	High
Performance	Moderate	High
Lexical Database	Extensive	Less extensive than WordNet
Integration	NLTK framework	spaCy pipeline
Ease of use	Requires POS tags	simple API, integrated POS tagging
Dependencies	Requires NLTK and WordNet databases	Requires spaCy models

WordNet Lemmatizer is well-suited for academic research and projects needing detailed lexical knowledge, despite its slower performance and the need for POS tags.
spaCy Lemmatizer excels in production environments and large-scale text processing due to its high performance and ease of integration within the spaCy pipeline.

Better to use WordNet for the project.
If Lemmatizing is used. no need of using stemming.

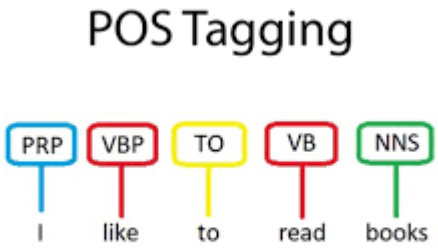
```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

def lemmatize_text(words): #words is a list of words
        return [lemmatizer.lemmatize(word) for word in words]

words = lemmatize_text(words)
```

Additional :-

Part-Of-Speech(POS) tagging in NLP



A fundamental preprocessing step in natural language processing that involves assigning a part of speech/words, to special tags.
The parts of speech include categories such as nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, and interjections.

Purpose,

- Helps in resolving ambiguity.
  Ex: the word "book" can be a noun ("I read a book") or a verb ("I will book a ticket")
- Provides a basis for syntactic parsing, which is essential for understanding the grammatical structure of sentences.
- Enhances the performance of various NLP applications such as information retrieval, named entity recognition, machine translation, and sentiment analysis.

POS tagging algorithms typically use two main approaches,

1. Rule based tagging : Uses a set of predefined rules to assign POS tags,.
   Ex: If a word ends in `"ing"`, it is often tagged as a verb (gerund).
2. Statistical tagging : Utilizes machine learning models and probabilistic techniques based on large annotated corpora.
   Ex:
   Hidden Markov Models (HMM),
   Maximum Entropy Models,
   Neural Networks

Popular POS tagging Algorithms

- Hidden Markov Models (HMM) - A statistical model that assumes the probability of a tag depends only on the previous tag.
- Maximum Entropy Models - A probabilistic model that considers various contextual features to predict the most likely tag for a word.

- Neural Networks - Deep learning models, such as recurrent neural networks (RNNs) and transformer-based models, have shown high accuracy in POS tagging by capturing complex patterns in text.

Tools for POS Tagging,

- NLTK
- spaCy
- Stanford NLP

Examples of POS Tagging

Consider the sentence,
"The quick brown fox jumps over the lazy dog."

Tokenization: `["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]`

POS tagging: `[("The", "DT"), ("quick", "JJ"), ("brown", "JJ"), ("fox", "NN"), ("jumps", "VBZ"), ("over", "IN"), ("the", "DT"), ("lazy", "JJ"), ("dog", "NN")]`

- DT: Determiner
- JJ: Adjective
- NN: Noun
- VBZ: Verb, 3rd person singular present
- IN: Preposition